REMARKS/ARGUMENTS

Claims 1-27 are pending in the application. Claims 1, 5, 6, 7, 10, 14, 15, 16, 19, 23, 24, and 25 have been amended. Claims 2, 11, and 20 have been cancelled. Reconsideration is respectfully requested. Applicants submit that the pending claims 1-27 are patentable over the art of record and allowance is respectfully requested of claims 1-27.

Applicants would like to thank Examiner Kiss for holding a telephone interview with their representative, Janaki K. Davda, on Wednesday, October 15, 2003. During the telephone interview, claims 1 and 2 were discussed with respect to the references.

The Office Action Summary indicates that the drawings submitted on February 29, 200 have been approved.

In paragraph 2, the Specification is objected to because on page 1, line 23, "280,371" should read --09/280,371–. Applicants' have amended the Specification to overcome the objection.

Additionally, in paragraph 3, the Specification is objected to due to the use of various trademarks. Applicants' have amended the Specification to overcome the objection.

In paragraph 5, claims 1-4, 6, 10-13, 15, 19-22, and 24 are rejected under 35 U.S.C. 102(b) as being anticipated by Ron Petrusha, "Inside the Windows 95 Registry" (hereinafter Petrusha).

Claims 1, 10, and 19 describe executing a command from an application program to store at least one variable maintained by the operating system in a data object accessible to the application program, wherein the application program is executing on the operating system. A command is received from an application program for at least one variable maintained by the operating system. It is determined whether the at least one variable is in a data object. See, for

example, Applicants' Specification at page 12, lines 26-27, and FIG. 3a, block 402. If the at least one variable is in the data object, the at least one variable is returned to the application program. See, for example, Applicants' Specification at page 12, lines 27-29, and FIG. 3a, blocks 404-406.

If the at least one variable is not in the data object, then the command from the application program is processed to retrieve and store the at least one variable in the data object. In particular, an operating system native command to use to retrieve the at least one variable is determined. For example, see Applicants' Specification, page 13, lines 2-5 and FIG. 3a, blocks 408-418. The operating system native command is executed in response to the command from the application program to retrieve the at least one variable into a buffer. For example, see Applicants' Specification, page 13, lines 11-13. The retrieved at least one variable is stored from the buffer into the data object. For example, see Applicants' Specification at page 13, lines 14-17, which indicates that environment variables are generated as a data stream, and this output data stream is captured and read to obtain the content of the variables. Applicants' Specification at page 13, lines 27-28, indicates that the content of the variables is added to the data object. The command from the application program is executed to retrieve the at least one variable from the data object for return to the application program. For example, see Applicants' Specification, page 14, lines 7-11 and FIG. 3b, block 440.

The Office Action cites the Petrusha reference on page 35 pages 38-41 as teaching the subject matter of claims 1, 10, and 19. The cited portion of the Petrusha reference describes a registry, which is a database, and a Registry Editor that provides a user interface for browsing the registry. There is no mention of retrieving a variable into a buffer and storing the retrieved variable from the buffer into a data object. Instead, the Petrusha reference describes exporting the registry to a .reg file and opening the .reg file with a text editor or word processor (page 41, second paragraph). Thus, the data in the registry is in a form that can be exported to a file that is read with a word processor, and this teaches away from storing variables in data objects.

Also, at pages 61-68, the Petrusha reference describes retrieving values (e.g., FindFlags and View entries, page 63) and provides code for displaying the user interface. Although the Registry Editor provides a user interface to access the registry, there is no description of, if a at least one variable is in a data object, returning the at least one variable to the application program, and, if the at least one variable is not in the data object, processing by command from the application program by determining an operating system native command to use to retrieve the at least one variable, retrieving a variable into a buffer, storing the variable from the buffer into a data object, and executing the command from the application program to retrieve the variable from the data object for return to the application program.

Therefore, claims 1, 10, and 19 are not anticipated by the Petrusha reference. Dependent claims 2-4, 11-13, 15, 18-22, and 24 incorporate the language of independent claims 1, 10, and 19 and add additional novel elements. Therefore, dependent claims 2-4, 11-13, 15, 18-22, and 24 are not anticipated by the Petrusha reference.

In paragraph 6, claims 1, 7, 8, 10, 16, 17, 19, 25, and 26 are rejected under 35 U.S.C. 102(e) as being anticipated by Steve DeGroof, "Class IniFile" (hereinafter DeGroof).

The DeGroof reference describes a class named "IniFile" for handling Windows-style INI files. The class includes a parseLines() method for reading lines, filling in subjects, variables and values. The DeGroof patent does not teach or suggest, if a at least one variable is in a data object, returning the at least one variable to the application program, and, if the at least one variable is not in the data object, processing by command from the application program by determining an operating system native command to use to retrieve the at least one variable, retrieving a variable into a buffer, storing the variable from the buffer into a data object, and executing the command from the application program to retrieve the variable from the data object for return to the application program. Instead, because the DeGroof patent has a method for directly reading the INI file, the DeGroof patent teaches away from Applicants' claimed subject matter.

Therefore, claims 1, 10, and 19 are not anticipated by the DeGroof reference. Dependent claims 7, 8, 16, 17, 25, and 26 incorporate the language of independent claims 1, 10, and 19 and add additional novel elements. Therefore, dependent claims 7, 8, 16, 17, 25, and 26 are not anticipated by the DeGroof reference.

In paragraph 7, claims 1, 5, 10, 14, 19, and 23 are rejected under 35 U.S.C. 102(e) as being anticipated by Jonathan Locke, "Parlex-vou J/Direct?" (hereinafter Locke).

The Locke reference describes J/Direct, which is a JAVA native interface. J/Direct is used to call a GetEnvironmentVariable function to fill in a string buffer value with the value of an environment variable. The Locke reference, however, does not describe storing the variable from the buffer into a data object and executing a command from an application program to retrieve the variable from the data object for return to the application program. For example, see Applicants' Specification at page 13, lines 14-17, which indicates that environment variables are generated as a data stream, and this output data stream is captured and read to obtain the content of the variables. Applicants' Specification at page 13, lines 27-28, indicates that the content of the variables is added to the data object.

Additionally, claims 1, 10, and 19 describe that if a at least one variable is in a data object, the at least one variable is returned to the application program. If the at least one variable is not in the data object, then an operating system native command to use to retrieve the at least one variable is determined, a variable is retrieved into a buffer, the variable from the buffer is stored into the data object, and the command from the application program is executed to retrieve the variable from the data object for return to the application program. On the other hand, J/Direct calls the GenEnvironmentVariable function each time that an environment variable is requested. Therefore, the Locke reference teaches away from Applicants' claimed subject matter.

Therefore, claims 1, 10, and 19 are not anticipated by the Locke reference. Dependent claims 5, 14, and 23 incorporate the language of independent claims 1, 10, and 19 and add

additional novel elements. Therefore, dependent claims 5, 14, and 23 are not anticipated by the Locke reference.

In paragraph 9, claims 9, 18, and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over DeGroof.

As discussed above with respect to claims 1, 10, and 19, because the DeGroof patent has a method for directly reading the INI file, the DeGroof patent teaches away from Applicants' claimed subject matter. Dependent claims 9, 18, and 27 incorporate the language of independent claims 1, 10, and 19 and add additional novel elements. Therefore, dependent claims 9, 18, and 27 are not taught or suggested by the DeGroof reference.

Conclusion

For all the above reasons, Applicant submits that the pending claims 1-27 are patentable over the art of record. Applicants have not added any claims. Nonetheless, should any additional fees be required, please charge Deposit Account No. 09-0447.

The attorney of record invites the Examiner to contact her at (310) 553-7973 if the Examiner believes such contact would advance the prosecution of the case.

Dated: October 17, 2003

By: Janaki K. Davda

Janaki K. Davda
Registration No. 40,684

Please direct all correspondences to:

David Victor
Konrad Raynes Victor & Mann, LLP
315 South Beverly Drive, Ste. 210
Beverly Hills, CA 90212
Tel: 310-553-7977
Fax: 310-556-7984